IN THE CLAIMS:

Please amend the claims as follows:

1. (Original) A computer system, comprising an output device and at least one processor which, when executing a debugging program, is configured to:

wait for a program being debugged to stop executing immediately prior to executing a next executable statement at which at least one variable has a current value; and

display on the output device the at least one variable in a manner that visually indicates an executable status of the at least one variable, wherein the executable status is indicative of at least one of a use and change of the current value during subsequent continuing execution of the program being debugged.

2. (Original) The system of claim 1, wherein the executable status indicates that the current value may change when the next executable statement is executed.

3. (Original) The system of claim 1, wherein the executable status indicates that the current value may be used when the next executable statement is executed.

4. (Original) The system of claim 1, wherein the executable status is visually represented on the output device to differentiate the at least one variable from other variables displayed on the output device.

5. (Original) The system of claim 1, further comprising a memory containing a monitor window interface configured to display the at least one variable on the output device in a manner to visually differentiate the at least one variable from other variables having a different executable status.

6. (Currently Amended) A method for displaying variables of a program being debugged, comprising:

when the program being debugged stops executing immediately prior to executing a next executable statement at which at least one variable has a current value, determining at least one of a first executable status and a second executable

status of the at least one variable based on a current point of execution, wherein the first executable status is defined by whether the current value of the at least one variable may change during subsequent execution of the program being debugged and the second executable status is defined by whether the current value of the at least one variable has a use during subsequent execution of the program being debugged; and

preparing an output which, when displayed on an output device, visually indicates ~~an~~ the executable status of the at least one variable at the current point of execution.

7. (Original) The method of claim 6, wherein the second executable status is defined according to one of only the next executable statement and any statement that may be encountered during subsequent execution of the program being debugged.

8. (Original) The method of claim 6, wherein the program being debugged stops executing upon encountering a breakpoint.

9. (Original) The method of claim 6, wherein determining comprises referring to a control flow graph.

10. (Original) The method of claim 6, wherein preparing comprises preparing the output so that, when displayed, the at least one variable is visually differentiable from other displayed variables according to their respective executable statuses.

11. (Original) The method of claim 6, wherein the first executable status indicates that the value of the at least one variable may change during subsequent execution of the program being debugged.

12. (Original) The method of claim 6, wherein the first executable status indicates that the value of the at least one variable may change during subsequent execution of the program being debugged and the second executable status indicates that the value of the at least one variable has a use during subsequent execution of the program being debugged.

13. (Original) The method of claim 6, wherein at least one variable is a variable referenced in the next executable statement in the program being debugged.

14. (Original) The method of claim 13, wherein the next executable statement contains a plurality of variables, and wherein preparing comprises preparing the output so that, when displayed, the at least one variable and the plurality of variables are visually differentiable from one another according to their respective different executable statuses.

15. (Original) The method of claim 6, further comprising displaying the variables with formatting to visually differentiate the variables from one another according to their respective executable statuses.

16. (Original) The method of claim 15, wherein the formatting comprises at least one of brackets, parentheses, asterisks, highlighting, strike-outs and numerals.

17. (Original) A method for displaying variables of a program being debugged, comprising:

when a program being debugged stops executing immediately prior to a next executable statement at which at least one variable has a current value, determining an executable status of at least one variable of the statement based on a current point of execution, wherein the executable status is indicative of at least one of a possible use and a possible change of the current value during subsequent continuing execution of the program being debugged; and

preparing an output which, when displayed on an output device, visually indicates the executable status of the at least one variable at the current point of execution.

18. (Original) The method of claim 17, wherein the executable status is defined according to one of only the next executable statement and any statement that may be encountered during the subsequent continuing execution of the program being debugged.

19. (Original) The method of claim 17, wherein the program being debugged stops executing upon encountering a breakpoint.

20. (Original) The method of claim 17, wherein a first executable status indicates that the value may change and a second executable status indicates that the value has a use.

21. (Original) The method of claim 17, wherein preparing comprises preparing the output so that, when displayed, the at least one variable is visually differentiable from other displayed variables according to their respective executable statuses.

22. (Original) The method of claim 17, wherein the next executable statement contains a plurality of variables, and wherein preparing comprises preparing the output so that, when displayed, the at least one variable and the plurality of variables are visually differentiable from one another according to their respective different executable statuses.

23. (Original) The method of claim 17, wherein the executable status indicates that the value of the at least one variable may have a use when execution of the program being debugged resumes and wherein preparing comprises preparing the output so that, when displayed, the variables are visually differentiable from one another according to their respective executable statuses.

24. (Original) The method of claim 17, wherein the executable status indicates that the value of the at least one variable may change when execution of the program being debugged resumes and wherein preparing comprises preparing the output so that, when displayed, the variables are visually differentiable from one another according to their respective executable statuses.

25. (Original) The method of claim 17, wherein determining comprises:

referring to a control flow graph to locate the next executable statement; and

accessing a variable-containing data structure associated with the next executable statement.

26. (Original) The method of claim 17, wherein determining further comprises:

referring to the control flow graph to locate an executable statement subsequent to the next executable statement; and

accessing a variable-containing data structure associated with the executable statement.

27. (Original) The method of claim 17, further comprising displaying the variables with formatting to visually differentiate the variables from one another according to their respective executable statuses.

28. (Original) The method of claim 27, wherein the formatting comprises at least one of brackets, parentheses, asterisks, highlighting, strike-outs and numerals.

29. (Original) A signal bearing medium, comprising a debugging program which, when executed by a processor, performs a method, comprising:

when a program being debugged stops executing immediately prior to a next executable statement at which at least one variable has a current value, determining an executable status of at least one variable of the statement based on a current point of execution, wherein the executable status is indicative of at least one of a possible use and a possible change of the current value during subsequent continuing execution of the program being debugged; and

preparing an output which, when displayed on an output device, visually indicates the executable status of the at least one variable at the current point of execution.

30. (Original) The signal bearing medium of claim 29, wherein the program being debugged stops executing upon encountering a breakpoint.

31. (Original) The signal bearing medium of claim 29, wherein determining comprises referring to a control flow graph.

32. (Original) The signal bearing medium of claim 29, wherein the executable status is defined according to one of only the next executable statement and any

statement that may be encountered during the subsequent continuing execution of the program being debugged.

33. (Original) The signal bearing medium of claim 29, wherein preparing comprises preparing the output so that, when displayed, the at least one variable is visually differentiable from other displayed variables according to their respective executable statuses.

34. (Original) The signal bearing medium of claim 29, wherein a first executable status indicates that the value may change and a second executable status indicates that the value has a use.

35. (Original) The signal bearing medium of claim 29, wherein determining comprises:

referring to a control flow graph to locate the next executable statement; and accessing a variable-containing data structure associated with the next executable statement.

36. (Original) The signal bearing medium of claim 35, wherein preparing comprises preparing the output so that, when displayed, the variables are visually differentiable from one another according to their respective executable statuses.

37. (Original) The signal bearing medium of claim 29, wherein the executable status indicates that the value of the at least one variable can change when execution of the program being debugged resumes.

38. (Original) The signal bearing medium of claim 37, wherein preparing comprises preparing the output so that, when displayed, the variables are visually differentiable from one another according to their respective executable statuses.

39. (Original) The signal bearing medium of claim 29, wherein the executable status indicates that the value of the at least one variable may have a use when execution of the program being debugged resumes and wherein preparing comprises

preparing the output so that, when displayed, the variables are visually differentiable from one another according to their respective executable statuses.

40. (Original) The signal bearing medium of claim 29, wherein the next executable statement contains a plurality of variables, and wherein preparing comprises preparing the output so that, when displayed, the at least one variable and the plurality of variables are visually differentiable from one another according to their respective different executable statuses.

41. (Original) The signal bearing medium of claim 29, further comprising displaying the variables with formatting to visually differentiate the variables from one another according to their respective executable statuses.

42. (Original) The signal bearing medium of claim 41, wherein the formatting comprises at least one of brackets, parentheses, asterisks, highlighting, strike-outs and numerals.